PATENT ABSTRACTS OF JAPAN

(11)Publication number : 04-051367
(43)Date of publication of application : 19.02.1992

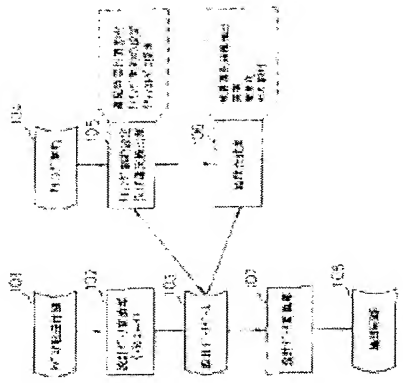
(51)Int.Cl. G06F 15/60

(21)Application number : 02-159948
(22)Date of filing : 20.06.1990

(71)Applicant : TOSHIBA CORP
(72)Inventor : NISHI HIROAKI

(54) SYNTHESIS DEVICE FOR LOGICAL CIRCUIT

(57)Abstract:
PURPOSE: To synthesize the logical circuits of high performance by providing a means which gives arbitrary delay time to the constitution element of an abstraction circuit, executing timing analysis at the stage of the abstraction circuit, and executing optimization so that the restriction of delay time is satisfied through the use of an analysis result.
CONSTITUTION: A timing restriction 104 sets restriction delay time in the arbitrary point of a register transfer specification. A timing restriction setting/ violation detection part 105 fetches data on the type of the element of an initial abstraction circuit from a design data base part 103, calculates the delay time of the element and allocates it to the initial abstraction circuit generated in a design data conversion part 102, fetches the timing restriction 104 and inspects whether given timing restriction is realized or not. A logical synthesis part 106 analyzes a violation circuit detected in the violation detection part 105, develops the abstraction circuit to a concrete circuit, simplifies a redundancy circuit and allocates data to the circuit element.



⑲日本国特許庁（JP）　⑪特許出願公開

⑫公開特許公報（A）　平4−51367

⑤1Int.Cl.⁵	識別記号	庁内整理番号	⑬公開　平成4年(1992)2月19日
G 06 F 15/60	360 K	7922−5L	

審査請求　未請求　請求項の数 2（全17頁）

⑤4発明の名称　論理回路の合成装置

⑳特　願　平2−159948
㉒出　願　平2(1990)6月20日

⑦2発明者　西　宏晃　神奈川県川崎市幸区小向東芝町1　株式会社東芝総合研究所内
⑦1出願人　株式会社東芝　神奈川県川崎市幸区堀川町72番地
⑦4代理人　弁理士　三好秀和　外1名

明　細　書

1．発明の名称

論理回路の合成装置

2．特許請求の範囲

（1）論理回路を自動生成するための論理回路の合成装置において、

合成の開始前、あるいはその途中で生成された抽象回路を使用してタイミング解析を行なうために、前記抽象回路を構成している素子に任意の遅延時間を与える手段と、

与えられた遅延時間を用いてタイミング解析を行い、タイミング制約条件が満たされるかどうかを検出する手段と、

前記検出されたタイミング解析結果を使用してタイミング制約条件が満たされるように前記抽象回路を論理回路に変換あるいは合成する手段と、

前記タイミング制約に違反する回路の遅延時間を最適化するために回路を変換あるいは合成する手段と、

を具備することを特徴とする論理回路の合成装置。

（2）前記遅延時間最適化のための回路の変換あるいは合成手段は、前記タイミング解析手段で検出された制約違反の回路を解析して前記回路を最適化対象回路に分割するとともに分割された回路に最適化目標値を設定し、前記最適化目標値にしたがって回路を変換あるいは合成する手段を有することを特徴とする請求項1記載の論理回路の合成装置。

3．発明の詳細な説明

［発明の目的］

（産業上の利用分野）

この発明は論理回路を自動的に合成するための装置に係り、特にタイミングの制約に基づいて論理回路を合成するための装置に関する。

（従来の技術）

論理合成技術の発展によって、レジスタ転送レベルの回路記述から構造レベルの回路を自動生成できるようになった。

ところが、熟練設計者が設計した論理回路と自

動合成された論理回路の性能（特に動作速度など）を比較すると、人手設計のほうが優れている。というのは、熟練設計者は、回路の動作周波数を満たすようにレジスタのクロックを決定し、クリティカルパスを絶えず考慮しながら論理回路を設計しているからである。すなわち、どの様な構造を持った回路にすれば信号遅延時間が少なくて済むかを熟知しているからである。

また設計者はタイミング解析ツールを使用することによって、論理回路の詳細な信号遅延時間の解析結果を入手できるようになったが（例えばどのパスが最長遅延もしくは最小遅延パスであるか）、これは設計者が所望する論理回路の設計を一通り終った後でないと使用できなかった。従って、論理合成後の論理回路に対してタイミング解析を行い、設計者が考えている制約に違反する箇所を人手で修正する必要があった。

一方、最近の論理合成技術では、ライブラリ化されたハードウェアコンポーネント（ライブラリセル）をゲート回路に割り当てた後で、論理合成

システムに内蔵したタイミング解析手段を用いてタイミング解析を実行し、この結果が設計者の所望するタイミング制約を満たしていない場合は、スピードの遅いハードウェアコンポーネントをスピードの速いハードウェアコンポーネントに交換し、再びタイミング解析を実行してタイミング制約が満たされているかどうかを調べ、これが満たされるまで、もしくはその限界まで繰り返すという手法をとっていた。

他方では、タイミング情報をハードウェアコンポーネントのテクノロジに依存しないライブラリ化された抽象コンポーネントに格納し、これを合成時に使用する手法も考えられている。この技術に関しては、特開昭６３－１５５２６８号公報において示されている。

しかし抽象コンポーネントは入力数が可変であったりコンポーネント自体ビット幅を有しているので、許されるビット幅に応じた膨大な数のコンポーネントをライブラリに登録する必要があり、従ってこの手法では多大な記録容量を必要として

いた。

（発明が解決しようとする課題）

上述したように従来技術では、部品ライブラリとして用意されたコンポーネントによって回路が構成されていないと、タイミング解析が実行できないという不具合があった。

また、レジスタ転送回路から変換された抽象回路では元来入出力数が可変のため、遅延時間を使用可能な全ての抽象回路に持たせて、それぞれをライブラリ化するとデータ量が膨大になり不経済である。

本発明は、このような点に関してなされたもので、その目的は、同一機能を有するが、しかし異なる構造を持ったハードウェアに適した遅延時間の割付手段を用いて抽象回路の段階でタイミング解析を実行し、その結果のタイミング違反情報を合成開始前あるいは途中の段階で使用することで高性能な論理回路を合成することが可能な装置を提供することである。

［発明の構成］

（課題を解決するための手段）

上記課題を解決するために本発明では、論理回路を自動生成するための論理回路の合成装置において、合成の開始前、あるいはその途中で生成された抽象回路を使用してタイミング解析を行なうために、前記抽象回路を構成している素子に任意の遅延時間を与える手段と、与えられた遅延時間を用いてタイミング解析を行い、タイミング制約条件が満たされるかどうかを検出する手段と、前記検出されたタイミング解析結果を使用してタイミング制約条件が満たされるように前記抽象回路を論理回路に変換あるいは合成する手段と、前記タイミング制約に違反する回路の遅延時間を最適化するために回路を変換あるいは合成する手段と、を具備することを特徴とする。

（作用）

本発明の装置によれば、機能動作仕様記述から変換された抽象回路を構成する素子に任意の遅延時間を割当てることができ、この遅延時間を使用して合成前あるいは合成途中にタイミング検証

が行える。さらにタイミング解析の結果を使用して遅延時間の制約を満足するように抽象回路を論理回路に変換または合成し最適化を繰返して行えるので、スピードの速い論理回路の合成ができる。

（実施例）

以下、本発明の実施例を図面を参照して説明する。

第１図は本発明の一実施例にかかる論理合成装置の構成を示すブロック図である。

第１図の１０１は機能動作仕様を表すレジスタ転送仕様の入力部で、レジスタ転送記述もしくは該仕様をスケマテック表示した機能ブロック図である。１０２は設計データ変換部（トランスレータ）である。この変換部１０２は入力部１０１のレジスタ転送記述を構文解析して、該記述に忠実な初期抽象回路を生成する。また機能ブロック図の場合はスケマテック図よりネットワーク情報と図形情報を抽出して初期抽象回路を生成する。１０３は該抽象回路のネットワーク情報を格納する設計データベース部である。１０４はタイミング

制約入力部で、レジスタ転送仕様の任意の点に制約遅延時間を設定できる。１０５はタイミング制約設定及び違反検出部で、トランスレータ部１０２で生成された抽象機能素子の入出力数が一定でないため、設計データベース部１０３から該初期抽象回路の素子のタイプと入力数またはビット幅を取り出し、該素子の遅延時間を計算し割り付ける遅延時間計算割付手段と、入力部１０４からタイミング制約を取り込む手段と、与えられたタイミング制約が、論理合成前及び途中の抽象回路もしくは論理回路において実現されるかどうかを検証するクリティカルパス探索手段から構成されている。１０６は論理合成部で、前記タイミング制約違反検出部１０５で検出された違反回路を解析して該回路を最適化対象回路に分割し、さらに最適化目標値を設定し、該最適化目標値に従って抽象回路を具象回路に展開し、また、冗長な回路の簡単化及び、テクノロジに依存した回路素子の割付けを行う各処理手段から構成されている。１０７は設計データ変換部で、記述変換やスケマテッ

ク図の生成を行うためのもので、具体的には設計データベースに格納されている回路情報を論理回路（構造記述またはスケマテック図）に変換する。１０８は合成された論理回路の出力部である。

第２図は、第１図で説明した論理合成装置の処理フローを説明する図である。

２０１は、レジスタ転送記述及びタイミング制約を入力するステップで、２０２は、記述の動作解析を行って初期抽象回路を生成するステップである。２０３は抽象回路素子に遅延時間を計算して割り付けるステップである。２０４は、割り付けた遅延時間を使用して全体回路の遅延時間計算を行い、クリティカルパスを検出しタイミング制約が守られているかどうかを調べるステップである。制約が守られていないときは、制約に違反している箇所をレポートもしくはスケマテック表示する。また、該タイミング制約に違反しているパスに関しては、この制約が実現されるようにスピードの速い回路に展開して最適化を行い（２０５）、ステップ２０３及び２０４を繰り返して該タイ

ミング制約が満たされているかどうかを調べ、ハードウェアコンポーネントを割り付ける（２０６）ことによって合成を終了する。以下各ステップの処理内容について詳細に説明する。

その前にレジスタ転送記述について説明する。レジスタ転送記述では、回路の素子と認識されるファシリティ（外部端子、内部信号、レジスタ、メモリなど）とハードウェアの詳細まで立ち入らずに機能を表せる演算子、組み込み関数（加算、減算など）、ファシリティの動作を表すプロセス（レジスタの非同期クリアなど）の記述や状態遷移表現が可能である。以下ファシリティ、演算子、組み込み関数、プロセス、ステートを抽象機能素子と呼ぶ。

ここで抽象機能素子の遅延時間について説明する。論理合成は最終的に抽象機能素子を論理素子（ゲート）まで展開して最適化を行い、これに所望のテクノロジからなるハードウェアのライブラリセルを割り付けるものである。しかしレジスタ転送レベルの回路は機能レベルの回路であり、必

ずしもハードウェアとして最適化された回路、すなわちゲート数とかゲート段数を考慮して設計された回路とはなっていない。従って最初からすべての抽象機能素子にハードウェアのセルを割り付けて考えるのは得策ではない。また同一機能のセルであっても異なるスピードのセルを割り付けることによって実遅延時間は変わってしまうので、レジスタ転送レベルの回路の各抽象機能素子に実時間を割り当てて信号伝播パスの遅延時間を計算するのは適当ではない。そこでこの実施例の装置では、基本的なゲートを単位としたゲート段数を遅延時間の代わりに用いる。またハードウェアのＮＯＲゲートのセルなどでは入力数が増加するにしたがって実遅延時間がかなり異なってくるので（例えば６入力のＮＯＲは２入力ＮＯＲの４倍の遅延時間を有する）、このようなゲートでは入力数を考慮して段数を与えることにする。

つぎに第１図の検出部１０５のサブ手段である遅延時間計算割付手段の処理について説明する。

まず論理合成途中で一部の抽象機能素子が具体

的なハードウェアセルに変換されてもクリティカルパスの検索が可能なように、ハードウェアセルのライブラリに存在するセルについては、それを構成する論理素子を用いて段数を計算し遅延情報のライブラリに格納する。またレジスタ転送レベルの抽象機能素子及び合成途中で生成される抽象機能素子または論理素子については、設計データベース部１０３から抽象機能素子を取り出して、該素子のタイプやビット幅や入力数を変数として、それらが論理合成途中に典型的な抽象素子や論理素子からなるネットワークに展開されるとした遅延モデルや、論理素子そのものの遅延モデルに従って段数を計算して、設計データベースに格納する。ここで抽象機能素子の遅延段数は抽象機能素子の展開後の回路によって異なってくるので、前記段数の計算を行う手段とともに、遅延モデルを複数用意しておく。またこの処理で一度計算した遅延段数については抽象機能素子のタイプ、入力数、ビット幅と一緒にワーク記憶領域に保存しておき、設計データベースから同一タイプの抽象機

能素子が再び取り出されたらハッシュイングを行って、該ワーク記憶領域のデータを取り出して割り付ける。

ここで第６図の１０ビット２入力全加算器６００を例にとって段数計算を説明する。

いま論理合成で加算器をリップルキャリー型の加算器に展開することになっている場合、第６図（ｂ）の回路に展開されるものとして遅延モデルを考える。第６図（ｂ）の回路は５個の２ビット２入力の全加算器６０１～６０５より構成されている。これらは下位ビットで計算されたキャリーの信号に対してシリアルに接続されており（６０１～６０５）、入力ＣＩからＣＯまでのパスが最大遅延段数を有している。また出力データＳ＜８：９＞（６０９）も該段数と等しい遅延段数を有している。従って２ビット２入力の加算器の最大遅延段数が６段であることがわかっているとき、１０ビット２入力全加算器の最大遅延段数は３０段になる。

以下第３図を用いて第１図の検出部１０５にお

けるクリティカルパス検索手段の処理について説明する。いま、クリティカルパスの探索域は入力－レジスタ間、入力－出力間、レジスター レジスタ間、レジスター出力間とする。

処理を開始する（３０１）。入力属性を有するファシリティやレジスタ転送記述に於ける定数に与えた初期遅延情報や終点の信号名およびタイミング制約データ（ゲート段数）情報を取り込む（３０２）。レジスタ転送記述をトランスレートした結果を格納した設計データベースからネットワークデータを抽出する（３０３）。ネットワークを構成している機能素子が遅延情報のライブラリ内に存在するかどうかを調べ（３０４）、存在する場合は該ライブラリよりその遅延情報を取り込み該機能素子に割り付ける（３０５）。存在しない場合は展開後の回路の遅延モデルに従い段数を計算して該機能素子に割り付ける（３０６）。つぎに入力点より終点信号方向に向ってパスをトレースする。ここで入力点とは入力属性や定数属性を有するファシリティのことで、終点とはレジ

スタのデータ入力や出力属性のファシリティを指す。このときデータパス、制御パス（レジスタ転送記述ではｉｆ、ｃａｓｅなどでデータ転送を制御でき、このｉｆ、ｃａｓｅの条件信号を作成するパス）、クロックパス（レジスタのクロック入力に接続されるパス）のそれぞれについてパスの有する最大段数を伝播しながら計算する。またクロックパスについてはレジスタに入力後はレジスタの出力に接続されているデータパスについて段数を計算する。一度計算したパス上の各素子の入力からの段数は設計データベースに格納する（３０７）。すべての終点に対する段数計算が終了したらつぎの処理を行う。

つぎに終点に割り付けられたタイミング制約情報を用いて、終点から入力点に向かって抽象機能素子が持っている段数を引去りながら、各抽象機能素子の出力で要求される最小の遅延段数（最もクリティカルな制約）を計算して設計データベースに格納する（３０８）。抽象素子間の信号はレジスタ転送レベルの回路ではファシリティとして

認識されるのでパス自体の遅延についても考慮される。さらにスラック（いま求めた制約の遅延段数とパスの入力からの段数との差）を計算して設計データベースに格納する（３０９）。ここでスラックがマイナス値（最大遅延を考えている）になった箇所はクルティカルパス上に存在する。

その後終点からスラックの値を頼りにタイミング制約を満足しないパスをトレースする（３１０）。さらにこのとき制御信号の値によって機能的に非現実的なパスをクリティカルパスの候補としている場合があるのでｉｆ、ｃａｓｅの条件が矛盾なく成立するパスを本当のクリティカルパスとし、設計データベースにパス情報を格納する。そしてクリティカルパス上の各抽象素子の出力における入力からの段数、制約、スラックなどの情報を設計データベースに格納して（３１１）本手段の処理を終了する（３１２）。

またクリティカルパス探索手段では、終点に与えた最大遅延段数制約に対するクリティカルパスmax（スラックmax）と同じように、最小遅延段

数制約に対するクリティカルパスmin（スラックmin）が同時に探索できる。

以下第４図を用いて第１図のクリティカルパス探索手段によって発見されたクリティカルパスから論理合成部１０６で行う被最適化回路（被最適化パス）の抽出と該パス上の抽象機能素子の選択、展開及び最適化処理の手順について説明する。

ここで説明の関係上本発明の論理合成が用いているルールベースの方式について簡単に説明する。レジスタ転送記述をトランスレータで初期回路に変換したあと、抽象機能素子をより抽象度の低い機能素子または論理素子に展開する。このとき展開の度合によってゲート数は多いがスピードの速い回路やスピードは遅いがゲート数は少ない回路が構成できるので、展開の制御はルールベース方式を用いている。すなわち展開の仕方をルールで選択できる。上述したように展開の仕方もルールで記述できるが展開ルールの選択制御そのものがルールで記述でき、このルールをメタルールと呼ぶことにする。また抽象機能素子同士を対象にし

た最適化はアルゴリズムを用いた手法では不可能なので、本最適化にもルールを用いている。

以下の実施例ではクリティカルパスmaxを最適化合成する手順について説明する。なお、今後特に断らない限りクリティカルパスはクリティカルパスmaxの意味で用い、スラックもスラックmaxの意味で使用する。クリティカルパスを解消するように合成するには、クリティカルパス探索手段によって計算されたスラック値をゼロにするようにすればよいのでスラック値の絶対値を最適化目標値とする。

処理を開始する（４０１）。クリティカルパスから被最適化パス及び最適化目標値を求める（４０２）。被最適化パスの選択については後述する。つぎに被最適化パス上の抽象機能素子から被展開候補抽象機能素子を捜し出し（４０３）（これについても後で詳しく説明する）、候補が存在する場合は、該抽象機能素子を展開したとき遅延段数が減少する（最適化目標値に近づく）かどうかを調べる（４０４）。なお遅延時間制付手段でレジ

スタ転送レベルの抽象機能素子には典型的な遅延段数を割り当ててあるので、スピードの速い抽象回路に展開するルールが存在すれば減少することがわかる。減少しない場合はつぎの候補の抽象機能素子を抽出する（４０３）。減少する場合は該抽象素子をより抽象度の低い抽象素子に展開する（４０５）。そして展開後の抽象素子に対して遅延情報ライブラリを用いて遅延段数を割り当てる。該ライブラリにない場合は計算して割り当てる（４０６）。展開後の抽象機能素子のネットワーク全体に対して段数計算を行い（４０７）、展開前の抽象機能素子の遅延段数といま計算した遅延段数を比較して、該被最適化パスの最適化目標値が達成されたかどうかを調べる（４０８）。達成されない場合は、つぎの候補の抽象機能素子がなくなるまで抽出して繰り返す。その後で最適化目標値が達成されない場合は、展開後の機能素子同士または被最適化パスの端点の機能素子同士について最適化を行う（４０９）。全ての被最適化パスに対して上記処理が終了したかどうか調べ（４

１０）、終了した場合は該全体回路に対してクリティカルパス検索を行い展開最適化合成前のクリティカルパスが解消されているかどうかを調べる（４１１）。解消されていない場合は一連の処理でクリティカルパスが解消されずに残るまでステップ４０２以下の処理を繰り返す（４１２）。クリティカルパスが解消された場合もしくは残存するのが確定した場合は、上記展開後の抽象機能素子以外のネットワークの抽象機能素子に対しても展開を行い、さらに素子数最適化を行う（４１３）。このときクリティカルパスの解消のときに展開した回路については素子数最適化の処理を行わない。以上で遅延時間最適化合成の処理を終了する（４１４）。

つぎに第５図を用いて第４図のステップ４０２の被最適化パスの抽出手順について説明する。処理を開始する（５０１）。まずクリティカルパス探索手段で発見されたクリティカルパスmax のスラック値から最も危険度の高い（余裕度の小さい）パスを選択する（５０２）。該パスが他のクリテ

ィカルパスと重複するかどうか調べ（５０３）、この重複パスがクリティカルパスmin でないことを確かめる（５０４）。クリティカルパスmin と重複していない場合、この重複クリティカルパスのスラックのうち危険度の最も小さい（余裕度の最も大きい）値を選ぶ（５０５）。そして該重複したサブパスを被最適パスとし、いま求めた危険度の最も小さい（余裕度の最も大きい）スラック値の絶対値を最適化目標値とする（５０６）。また非重複サブパスの最適化目標値は、各クリティカルパスが有するスラック値に重複サブパスの最適化目標値を加えた値の絶対値になる（５０７）。ただしこの値は重複サブパスの最適化が成功したときに用いる。すなわち最適化に失敗した場合は該重複クリティカルパスの未達最適化目標値分をそれぞれ被最適化サブパスの最適化目標値に対して加算し、最適化目標値を更新する。ステップ５０３で調べたクリティカルパスが他のものと重複しない場合は、取り出したクリティカルパス全体を被最適化パスとし、またスラック値より最適化

目標値を求める（５０８）。すべてのクリティカルパスを被最適化パスに分けその最適化目標値が求まったら（５０９）本処理を中止し（５１０）、論理合成へデータをインターフェースする。すなわち処理４０２へ進む。

つぎに第４図のステップ４０３の被最適化パス上の抽象機能素子の選択手順について説明する。

まず、（１）外部出力に最も近い抽象機能素子を捜し出す。これは該抽象素子の変換によって発生する他の回路への影響を最小限にするためである。

（２）最も小さいファンアウトを有する抽象機能素子を捜し出す。これは、いまファンアウトが多い抽象素子を同一機能のより小さい遅延段数をもつ抽象素子に変換したとき、該変換後の抽象素子をノードとして持つパスが最小段数違反パスとなる可能性があるからである。

（３）最も小さいファンインを有する抽象機能素子を捜し出す。これは、いまファンインが多い抽象素子を同一機能のより小さい遅延段数をもつ抽

象素子に変換したとき、該変換後の抽象素子をノードとして持つパスが最小段数違反パスとなる可能性があるからである。

前記（１）～（３）の優先度は（２）と（３）を満たすものでかつ（１）を満足するものが最優先で、（２）と（３）に関しては（２）の値のより小さい抽象機能素子を選択する。この優先度についてはルールベース方式を用いているので自由に変更できる。

ここで第６図を用いて第１図の論理合成部１０６における抽象機能素子の展開の実施例について説明する。

第６図（ａ）に示す１０ビット２入力加算器６００の展開の手順について説明する。第６図（ｂ）は、２ビット加算器を５つ（６０１～６０５）用いてリップルキャリー１０ビット２入力加算器を構成した図である。第６図（ｃ）は２ビット加算器１つ（６０６）と４ビット加算器２つ（６０７，６０８）を用いて１０ビット２入力加算器を構成した図である。２ビット加算器のゲート数は２０

でゲート段数は３段である。一方４ビット加算器のゲート数は４５で段数は５段である。そこで第６図（ｂ）の構造にするとゲート数は１００でゲート段数は１５段になる。一方、第６図（ｃ）の構造にするとゲート数は１１０となりゲート段数は１２段となる。

いま、クリティカルパス上の１０ビット加算器を高速回路に展開する要求が出されているときは、前記メタルールを用いて第６図（ｃ）の回路を選択しこれに展開する。また反対に１０ビット加算器がクリティカルパス上にない場合は、メタルールを用いて第６図（ｂ）の回路を選択してこれに展開する。

展開後の抽象機能素子の最適化については、いま第６図（ｂ）の様な回路がクリティカルパス上に存在する場合、２ビット加算器６０１と６０２の２つを４ビット加算器１つに置き換えることによって段数が６段から５段に減少し１段数分スピードが速くなる。

つぎに第７図を用いて抽象機能素子から論理素

子への変換について説明する。

いま８ビットの１加算器（インクリメンタ）を例に取って説明する。レジスタ転送レベル言語で記述された８ビットの１加算器は、ＩＮＣ７０１という抽象素子にトランスレートされる。そこで論理構成の処理が開始されるとＩＮＣ７０１という素子はＥＸＯＲ（排他的論理和）７０２、ＡＮＤ７０３、ＮＯＴ７０４の論理素子に展開される。いま第７図（ｂ）の論理素子からなる回路と、第７図（ｃ）の回路と２つが考えられ、第７図（ｂ）の回路では各ビットのキャリーの計算を独自の回路で行っているのに対して、第７図（ｃ）の回路は４ビットの１加算器を２つ用いた構成であり、したがって下位４ビットからのキャリーの計算結果を用いるようになっている。２つの展開回路の違いはそれぞれ他の物と比較して、第７図（ｂ）では信号伝播遅延が小さいのに対して（ゲート数４８、ゲート段数４段）、第７図（ｃ）ではゲート数が小さい（ゲート数４０、ゲート段数５段）という特徴がある。この展開の処理をルールベー

スで行う場合、まず２つの基本展開ルールが用意されており、前記クリティカルパス探索手段で発見されたクリティカルパス上にＩＮＣと言う素子が使用されており、遅延最小の条件で展開の要求が出された場合は、メタルールでそれを判断して第７図（ｂ）の回路に展開し、さもなければ第７図（ｃ）の回路に展開する。

さらに簡単化ルールとして４入力ＡＮＤと２入力ＡＮＤがシリアルに接続されている場合、３入力ＡＮＤ２つに変換するというものがある場合、上記第７図（ｂ）の回路は第７図（ｃ）の回路になることが考えられるが、ゲート数最小化のために設けられたＡＮＤのシリアル結合は簡単化しないようにフラグをつけておく。そうすることによって、遅延時間最小化制約を満たす構造回路を生成できる。

上記実施例は、遅延時間最小化のルールによって生成された回路が、上記簡単化ルールのようなゲート数最小化のルールによって回路が変更されないようにした例であるが、反対にゲート数最小

化のルールによって回路を小さくする場合はフラグをｏｆｆにすればよい。

以下簡単なモチーフを用いてクリティカルパスの探索と合成の処理を説明する。

第８図は合成前の機能ブロック図である。８０１～８０５は入力端子である。ここで入力端子８０１～８０３は４ビット幅を有している。８０６，８０７はクロック入力端子である。また８０８は４ビット幅の出力端子である。８０９は４ビット２入力加算器で、８１０は４ビット２入力減算器である。８１１は比較器で、８１２は４ビットの１加算器（ＩＮＣ）、８１３は４ビットの１減算器（ＤＥＣ）である。８１４と８１５はそれぞれ１ビット、４ビットのレジスタで、８１６と８１７は２－１のマルチプレクサである。さらに８１８はＡＮＤで、８１９，８２０はインバータである。また８２１は内部信号である。

まずこの回路の機能素子及び結線情報をスケマテック入力または機能設計記述言語で書く。この情報をデータ変換手段を用いて回路設計データに

変換する。いまこの回路全体を論理合成の対象として遅延時間の制約を与える。ここでは、レジスタＲ２（８１５）を終点とするパスに対して最大遅延制約段数１０を与える。また出力端子Ｇ（８０８）に最大遅延制約段数１９を設定する。この設定の仕方はスケマテック入力ではレジスタシンボルＲ２のデータ入力端子に対して最大遅延時間制約のテキスト入力テーブルを開いて行うことができる。出力端子Ｇについても同様である。スケマテックの情報がなくてもこれらの制約はコマンドで入力できる。

つぎに各抽象機能素子に対する遅延情報の与え方について説明する。設計者は機能ブロックの遅延時間を自由に選択または設定できることが重要だが、機能ブロックをスケマテック入力するたびに、もしくは機能素子を記述するたびに遅延時間を設定していたのでは設計効率が大変悪いので、自動割当の手段を用いる。抽象機能素子の遅延時間は入力数及びそのビット数に依存し、さらに同一機能に対して構造回路は複数考えられるため、

ここではゲート数も遅延時間も良好な標準的な論理回路を想定して遅延時間を与える。いま第８図の点線の円で囲まれた数字は各機能ブロックの最大遅延段数である。インバータは他の論理ゲートに比べて遅延時間が小さいのでここではゲート段数を０としている。

これを整理して示すと以下の表１のようになる。

表　１

機能ブロックタイプ	遅延段数
ＡＤＤ	6
ＳＵＢ	7
ＣＯＭＰ	5
ＭＵＸ	1
ＩＮＣ	3
ＤＥＣ	3
ＲＥＧ	2
ＡＮＤ	1
ＩＮＶ	0

これらは設計データベースから各抽象機能素子のタイプと入力数と入力数のビット幅を取り出し

て、この情報をもとに計算して割り当てたものであり、これらについては合成途中で再度生成される可能性があるためワーク記憶領域に保存しておく。

そこで上で設定した遅延時間制約に基づいたクリティカルパス探索の処理について説明する。

入力端子からの各抽象機能素子の出力における段数を求める。入力点Ａ（８０２）、Ｂ（８０３）、Ｃ（８０４）、Ｄ（８０５）は０段で、クロック入力ＣＫ１（８０６）にはＡ～Ｄ入力に対して遅延時間として３段遅延を持たせ、ＣＫ２（８０７）は１６段とする。

いま入力端子Ａ（８０２）、Ｂ（８０３）からのデータパスの段数について説明する。点線の四角形の上から１段目の値が入力からの段数を示す。減算器（８１０）の出力は７段となり加算器（８０９）の出力は６段となる。またＭＵＸ（８１６）の出力は偽入力（８１０）からの段数が大きいのでそれをとり８段となる。一方ＭＵＸ（８１６）の制御入力はＣＫ１（８０６）が３段でレジスタ

Ｒ１（８１４）の段数２を加えて５段となる。またＭＵＸ（８１６）の出力は最大６段となる。従ってこのパスはデータパスからの８段をとる。比較器（８１１）の出力はＦ（８２１）からのパスの８段と比較器本体が有している５段を加えた１３段となる。さらにＭＵＸ（８１７）の出力はデータパスＦ（８２１）や１加算器（８１２）の出力段数の３段よりも大きい１４段の値をとる。最終的に、レジスタＲ２（８１５）のデータ入力へのパスの最大段数は１４段となる。

つぎにＲ２（８１５）を終点とするパスに対して制約段数（点数の四角形の２段目の値）を計算する。Ｒ２には１０段の最大遅延の制約が割り付けられている。Ｒ２へ入力するパスはＭＵＸ（８１７）から来ている。ＭＵＸの右から入力する制御信号８２２は偽のときの条件を表しており、８２３は真のときの条件を表している。ＭＵＸ（８１７）の左から入力する信号はデータ信号で右から入力する制御信号に対応するものである。いまＭＵＸのデータ入力Ｆ（８２１）に於ける制約を

求めると９段になる。ここでＦはファンアウトが２以上であるのでＦの出力のもとの遅延段数決めることができない。そこでＭＵＸ（８１７）の真入力のデータパスの制約を調べると、偽の場合と同様に９段となり、さらにＩＮＣ（８１２）は３段なのでＩＮＣの入力は６段となる。このパスについては、いま終点を指定したレジスタの出力に達したのでここで終了する。そこでＭＵＸ（８１７）の制御信号のパスの制約段数を求める。ＭＵＸの制御入力端子はそれぞれ９段である。ＭＵＸの偽のパスのインバータによる遅延は考慮せず比較器（８１１）の出力端子に於ける段数は９段となる。そして比較器の入力端子に於ける段数を計算する。比較器の遅延段数は５段であるので入力Ｆ（８２１）の出力は先ほど計算した９段よりも小さい、いま求めた４段を制約とする。以下同様に加算器（８０９）、減算器（８１０）の出力はそれぞれ３段となり減算器（８１０）の遅延段数が７段であるのでＡ（８０２）、Ｂ（８０３）はそれぞれ－４段となる。一方ＭＵＸ（８１６）の

制御入力に対してレジスタＲ１（８１４）の出力は３段となり、レジスタＲ１（８１４）のクロック入力からデータ出力までの遅延段数を差し引いてクロック入力ＣＫ１（８０６）は１段となる。

ここでスラック（制約段数－始点からの段数）を求める。点線の３段組の長方形の３段目の値がスラック値である。そしてＲ２（８１５）からこのスラックの値の内マイナス値をトレースすればこれがクリティカルパスとなる。このモチーフでは－４のパスが最も危険なパスで、つぎが－３のパスである。

同様に出力Ｇ（８０８）については、まず入力ＣＫ２（８０７）からの段数は入力Ａ（８０２）、Ｂ（８０３）、Ｅ（８０１）に対して１６段の遅延を持っているので、レジスタＲ２（８１５）のクロック入力では１６段となり、Ｒ２のデータ出力では１８段となる。さらにＤＥＣ（８１３）の出力端子で２１段となり結局出力Ｇ（８０８）で２１段になる。つぎに出力Ｇには制約１９段が与えられているので、ＤＥＣ（８１３）の入力はＤ

ＥＣ自体が有している３段の遅延段数を差し引き１６段となる。またレジスタＲ２のクロック入力からデータ出力までの遅延段数を差し引いてクロック入力ＣＫ２（８０７）は１４段となる。スラックは－２となりクリティカルパスはＣＫ２－Ｒ２－ＤＥＣ－Ｇのパスとなる。

つぎに最適化パスの選択方法について説明する。クリティカルパスが有しているスラック値から最も危険度の高いパスを選ぶ。いまＲ２のデータ入力に与えた制約に対して危険度の高いパスから示す。ｃｐ１．１とｃｐ１．２はスラックが－４のクルティカルパスで、ｃｐ２．１とｃｐ２．２はスラックが－３のクリティカルパスで、ｃｐ３．１は－２のクリティカルパスである。個々のクリティカルパスの最適化目標値はスラック値の絶対値としｔａｒｇｅｔ１～３に示す。以上を総合すると次の第２表のようになる。

（以下余白）

第２表

ｃｐ１．１：Ａ－ＳＵＢ－ＭＵＸ－Ｆ－ＣＯＭＰ－ＭＵＸ－Ｒ２
ｃｐ１．２：Ｂ－ＳＵＢ－ＭＵＸ－Ｆ－ＣＯＭＰ－ＭＵＸ－Ｒ２
ｓｌａｃｋ１：－４
ｔａｒｇｅｔ１：４
ｃｐ２．１：Ａ－ＡＤＤ－ＭＵＸ－Ｆ－ＣＯＭＰ－ＭＵＸ－Ｒ２
ｃｐ２．２：Ｂ－ＡＤＤ－ＭＵＸ－Ｆ－ＣＯＭＰ－ＭＵＸ－Ｒ２
ｓｌａｃｋ２：－３
ｔａｒｇｅｔ２：３
ｃｐ３．１：ＣＫ１－Ｒ１－ＭＵＸ－Ｆ－ＣＯＭＰ－ＭＵＸ－Ｒ２
ｓｌａｃｋ３：－２
ｔａｒｇｅｔ３：２

いま、ｃｐ１．１に着目するとこれは他の全てのクリティカルパスに重複する部分を有しており、

これを重複パスｏｖｐ１とする。最適化目標値は全クリティカルパスのうちもっとも危険度の小さいスラックの絶対値を取るので２（ｏｔ１）となる。またｃｐ１．１はｃｐ１．２とも重複しておりこれをｏｖｐ２と示す。最適化目標値はｔａｒｇｅｔ１－ｏｔ１で２（ｏｔ２）となる。
ｏｖｐ１：ＭＵＸ－Ｆ－ＣＯＭＰ－ＭＵＸ－Ｒ２
ｏｔ１：２
ｏｖｐ２：ＳＵＢ－ＭＵＸ
ｏｔ２：２

つぎにｃｐ２．１に着目すると重複パスｏｖｐ１について処理済みなので、ｃｐ２．２との重複パスｏｖｐ３を被最適化パスとし、最適化目標値ｏｖ３は１（ｔａｒｇｅｔ２－ｏｔ１）となる。ｃｐ３．１についてはｏｐｐ１が被最適化パスの候補になるが最適化目標値ｏｐ１は０（ｔａｒｇｅｔ３－ｏｔ１）となるため候補からはずす。
ｏｖｐ３：ＡＤＤ－ＭＵＸ
ｏｖ３：１
ｏｐｐ１：ＣＫ１－Ｒ１－ＭＵＸ

ｏｐ１：０

但しｏｐｐ１はｏｖｐ１で最適化が実現できた場合にのみ最適化目標値はゼロとなる。

つぎに被最適化パスｏｖｐ１について適用する最適化手法について説明する。

いま説明する手法は被最適化パス上の複数の機能素子に対して応分の最適化目標値を分担するものである。一方被最適化パス上のある機能素子に対して最大限の最適化目標値を担わせ、最小回数の展開で目標値を実現する手法でも実現できる。

まずはじめに、よりファンアウトの小さい抽象機能素子を選択するがその対象となるものはＭＵＸ８１７である。そこでＭＵＸの展開方法について高速化ルールが存在するかどうか調べ、いま高速化回路に展開するルールが存在しないので（本実施例では存在しないとした）つぎの候補を捜す。またファンアウトの小さい抽象機能素子を選択するとＣＯＭＰ（８１１）が候補に挙がる。そこでＣＯＭＰを展開したあとで高速化されるかどうか調べると是であるので実際に展開を行う。展開後

の回路を第９図に示す。そして展開後の個々の機能素子に対して遅延段数を与える。展開後の部分回路の段数計算を行うと４段になるので１段分だけ最適化目標値が達成された。ただしＥＸＯＲ（９０１）は２段、４入力ＮＯＲ（９０２）は２段としている。ところが被最適化パスｏｖｐ１は最適化目標値が２段であるので、まだ上記展開のみでは達成されないためつぎの候補を捜しにいくが、ＭＵＸ（８１６）も上記理由により最適化展開が不可能である。そのためＣＯＭＰの展開後の回路について機能素子間で遅延段数の最適化を行うが、第９図のように最適化の余地が残っていないのでつぎの被最適化パスの展開を行う。

被最適化パスｏｖｐ２について最適化目標値を計算する。重複したクリティカルパスｏｖｐ１の最適化目標値が実現できなかったので、本最適化パスはその分最適化目標値が大きくなる。本来の最適化目標値は２段であるが、いまｏｖｐ１の非実現最適化目標値１段を加えて３段になる。そこで最適化展開候補を捜すとＳＵＢ（８１０）が選

択され、さらに高速化回路に展開するルールがあるかどうかを調べると存在するので、ＳＵＢを高速化回路に展開する。ここでは５段の遅延段数を持つ回路に展開できるので（第１０図を参照、１００１は４ビット全加算器で遅延段数は５段）、最適化目標値３に対して２段分を満足させる。ｏｖｐ２には他に最適化展開可能な機能素子がないので、展開後の回路について機能素子間の最適化を行うが、最適化の余地が残っていないのでクリティカルパスｃｐ１．１とｃｐ１．２は最適化目標値４を達成できず３段分しか稼げなかったことになる。同様にｏｖｐ３の最適化目標値は２段になりパス上のＡＤＤは展開後最小５段にしかならないのでクリティカルパスｃｐ２．１、ｃｐ２．２の最適化目標値の３段は実現できず１段分の危険度は残る。さらに重複クリティカルパスの最適化目標値が実現できなかったことから、パスｏｐｐ１は最適化目標値１段を持った被最適化パスになるが、ｏｐｐ１上にはＭＵＸとかＲＥＧ等最適化候補がないためなにも行われない。従ってクリ

ティカルパスｃｐ３．１について１段分の危険度が残ってしまう。また各被最適化パスの端点同士の最適化が可能かどうか試みるが本実施例では被最適化パスがＭＵＸ（８１６）で分離されているため不可能である。本実施例によると最長遅延段数が１４段から１１段に削減された。

本実施例は展開後の回路の遅延段数を既知として説明したが、未知の場合は展開後の回路の段数計算を行う必要がある。さらに展開後の回路同士で遅延段数の最適化が行われた場合は再度クリティカルパス探索手段で展開前のクリティカルパスが回避されたかどうか調べ、クリティカルパス上の機能素子がすべて展開されるまで繰り返す。

［発明の効果］

以上説明したように、本発明の装置によれば少ないメモリ容量で抽象回路のレベルでタイミング検証が行え、合成前にタイミング仕様が満たされるか否かの判定が可能になる。また、抽象回路に割り当てられた遅延時間を利用して、回路の展開及び最適化時にスピードの速い回路を生成するの

で、論理式の簡単化やハードウェアのインプリメンテーションによる遅延時間の最適化だけを用いた場合よりも遅延時間の制約を満足する回路が実現できるという効果がある。

４．図面の簡単な説明

第１図は本発明の一実施例にかかる論理回路の合成装置の構成を説明するブロック図、

第２図は本発明の一実施例における論理合成の処理フローを説明する図、

第３図は本発明の一実施例におけるクリティカルパス探索手段の処理を説明する図、

第４図は本発明の一実施例における被最適化回路（被最適化パス）の抽出と該パス上の抽象機能素子の選択、展開及び最適化処理の手順について説明する図、

第５図は本発明の一実施例におけるクリティカルパスから被最適化パスの抽出手法について説明する図、

第６図は本発明の一実施例における抽象機能素子の展開について説明する図、

第７図は本発明の一実施例における抽象機能素子から論理素子への変換について説明する図、

第８図は本発明の一実施例におけるクリティカルパスの探索と合成の処理を説明するのに使用する簡単なモチーフの機能ブロック図、

第９図は本発明の一実施例における比較器の論理素子への変換について説明する図、および

第１０図は本発明の一実施例における減算器の展開について説明する図である。

１０１…レジスタ転送仕様の入力部

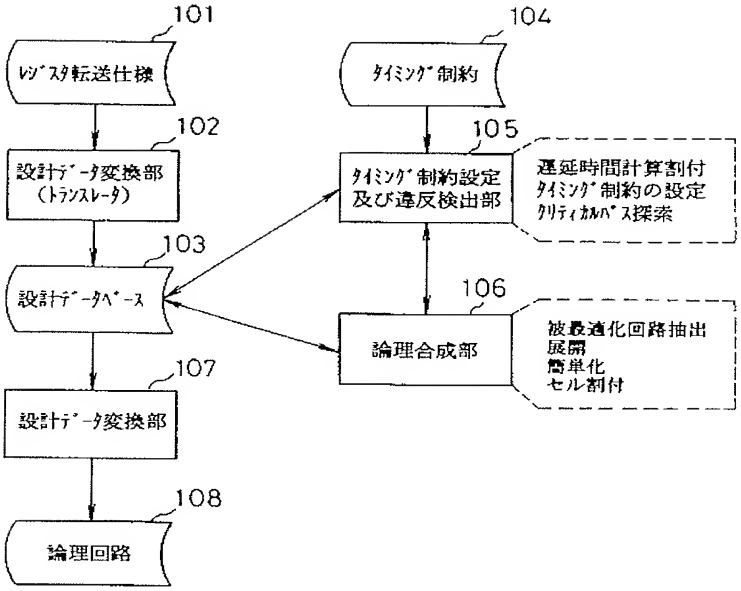
１０４…タイミング制約の入力部

１０５…タイミング制約設定及び違反検出部

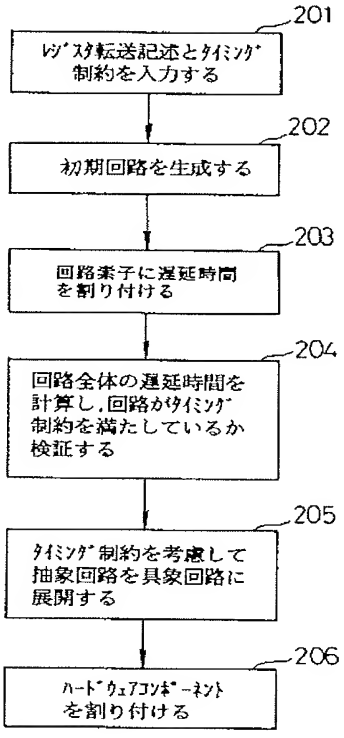
１０６…論理合成部

１０８…論理回路の出力部

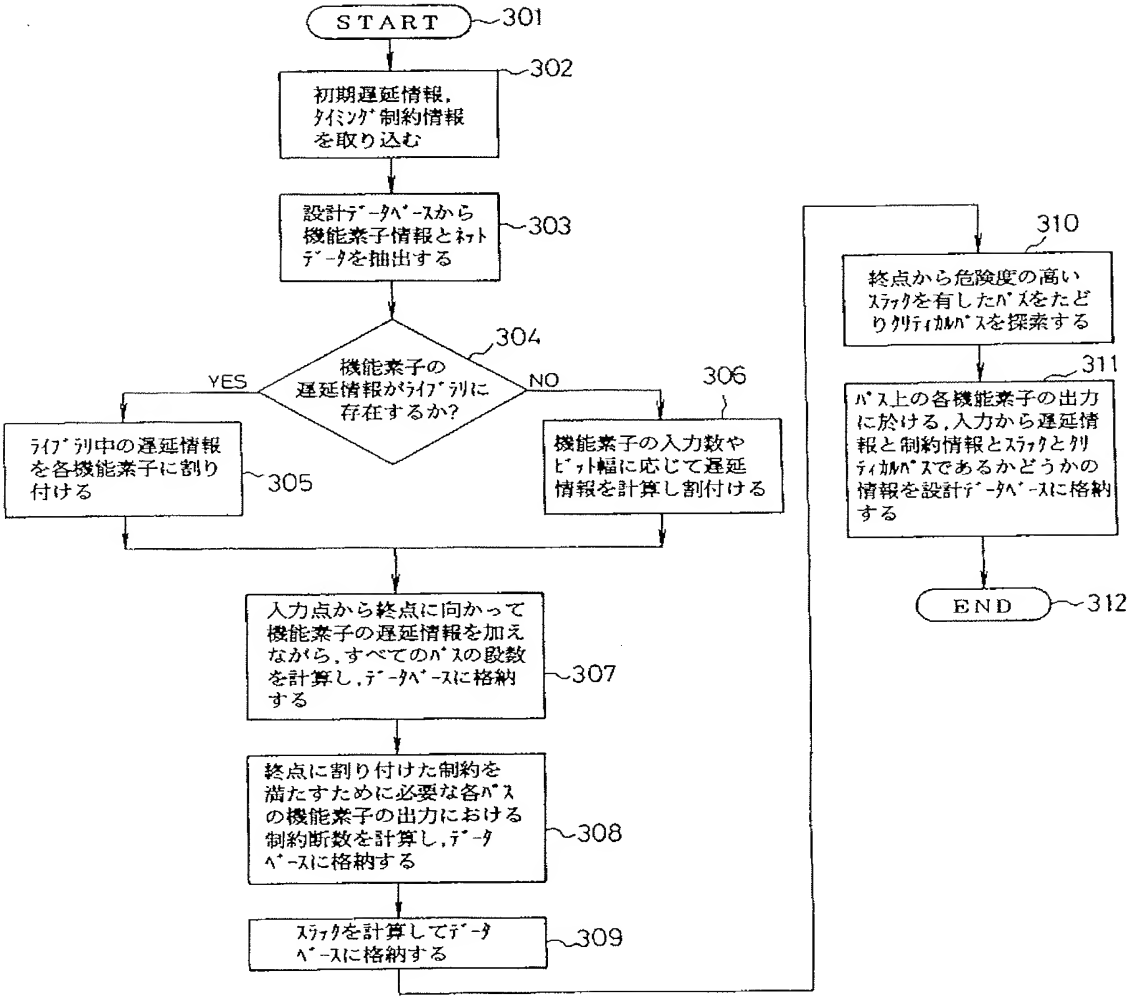
代理人弁理士 三好秀和


101
ﾚｼﾞｽﾀ転送仕様
102
設計ﾃﾞｰﾀ変換部
(ﾄﾗﾝｽﾚｰﾀ)
103
設計ﾃﾞｰﾀﾍﾞｰｽ
107
設計ﾃﾞｰﾀ変換部
108
論理回路
104
ﾀｲﾐﾝｸﾞ制約
105
ﾀｲﾐﾝｸﾞ制約設定
及び違反検出部
遅延時間計算割付
ﾀｲﾐﾝｸﾞ制約の設定
ｸﾘﾃｨｶﾙﾊﾟｽ探索
106
論理合成部
被最適化回路抽出
展開
簡単化
セル割付

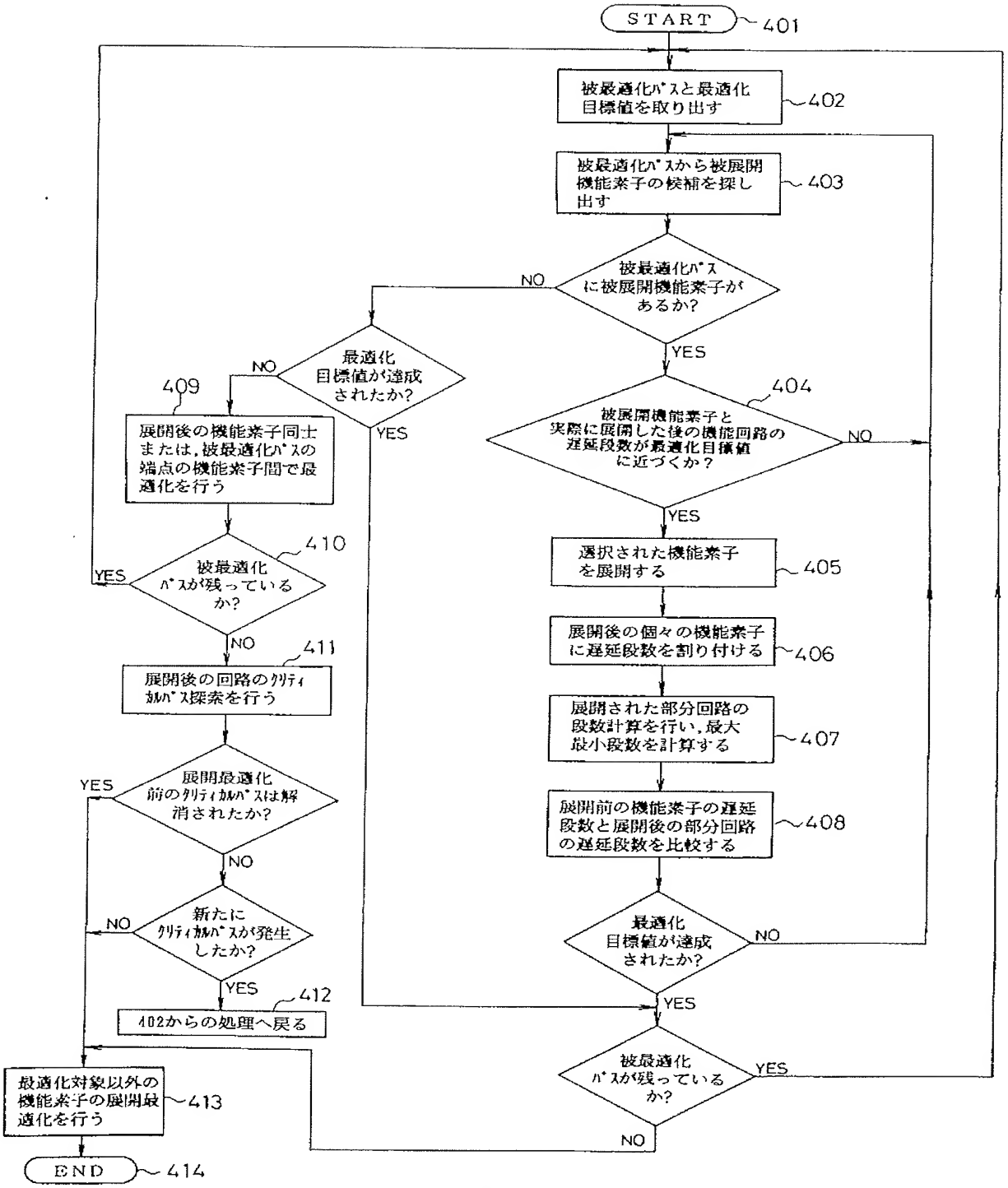
第 1 図


201
ﾚｼﾞｽﾀ転送記述とﾀｲﾐﾝｸﾞ
制約を入力する
202
初期回路を生成する
203
回路素子に遅延時間
を割り付ける
204
回路全体の遅延時間を
計算し，回路がﾀｲﾐﾝｸﾞ
制約を満たしているか
検証する
205
ﾀｲﾐﾝｸﾞ制約を考慮して
抽象回路を具象回路に
展開する
206
ﾊｰﾄﾞｳｪｱｺﾝﾎﾟｰﾈﾝﾄ
を割り付ける

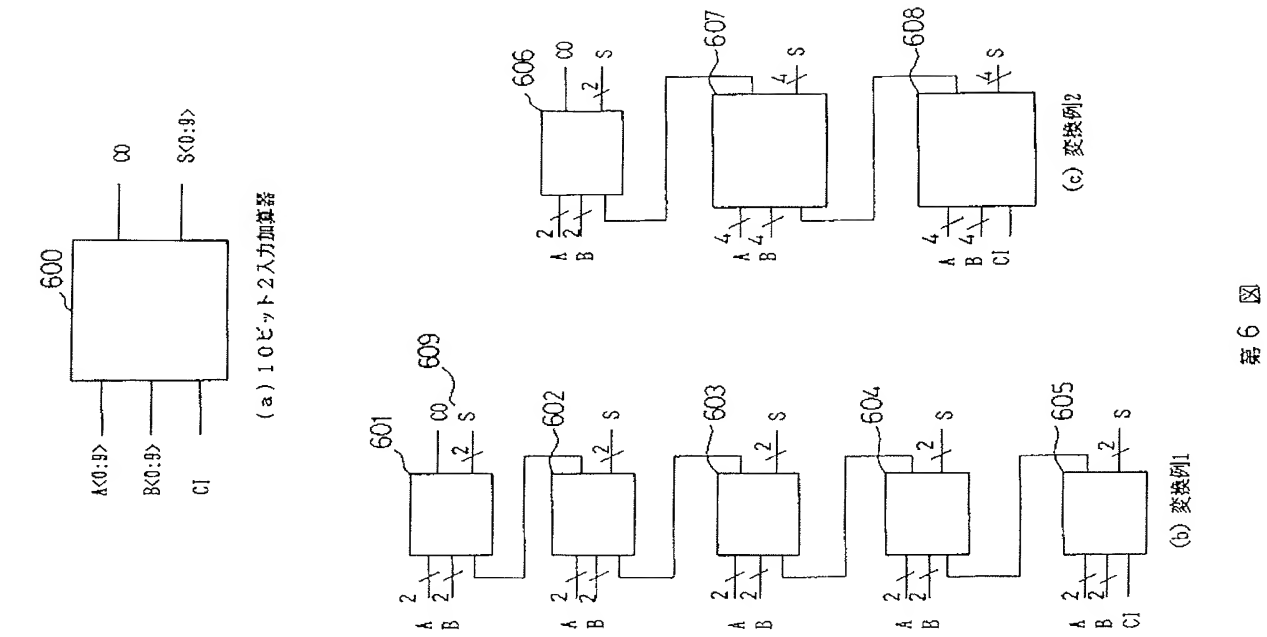
第 2 図


START
301
302
初期遅延情報，
ﾀｲﾐﾝｸﾞ制約情報
を取り込む
303
設計ﾃﾞｰﾀﾍﾞｰｽから
機能素子情報とﾈｯﾄ
ﾃﾞｰﾀを抽出する
304
機能素子の
遅延情報がﾗｲﾌﾞﾗﾘに
存在するか?
YES
NO
306
ﾗｲﾌﾞﾗﾘ中の遅延情報
を各機能素子に割り
付ける
305
機能素子の入力数や
ﾋﾞｯﾄ幅に応じて遅延
情報を計算し割付ける
入力点から終点に向かって
機能素子の遅延情報を加え
ながら，すべてのﾊﾟｽの段数
を計算し，ﾃﾞｰﾀﾍﾞｰｽに格納
する
307
終点に割り付けた制約を
満たすために必要な各ﾊﾟｽ
の機能素子の出力における
制約断数を計算し，ﾃﾞｰﾀ
ﾍﾞｰｽに格納する
308
ｽﾗｯｸを計算してﾃﾞｰﾀ
ﾍﾞｰｽに格納する
309
310
終点から危険度の高い
ｽﾗｯｸを有したﾊﾟｽをたど
りｸﾘﾃｨｶﾙﾊﾟｽを探索する
311
ﾊﾟｽ上の各機能素子の出力
に於ける，入力から遅延情
報と制約情報とｽﾗｯｸとｸﾘ
ﾃｨｶﾙﾊﾟｽであるかどうかの
情報を設計ﾃﾞｰﾀﾍﾞｰｽに格納
する
END
312

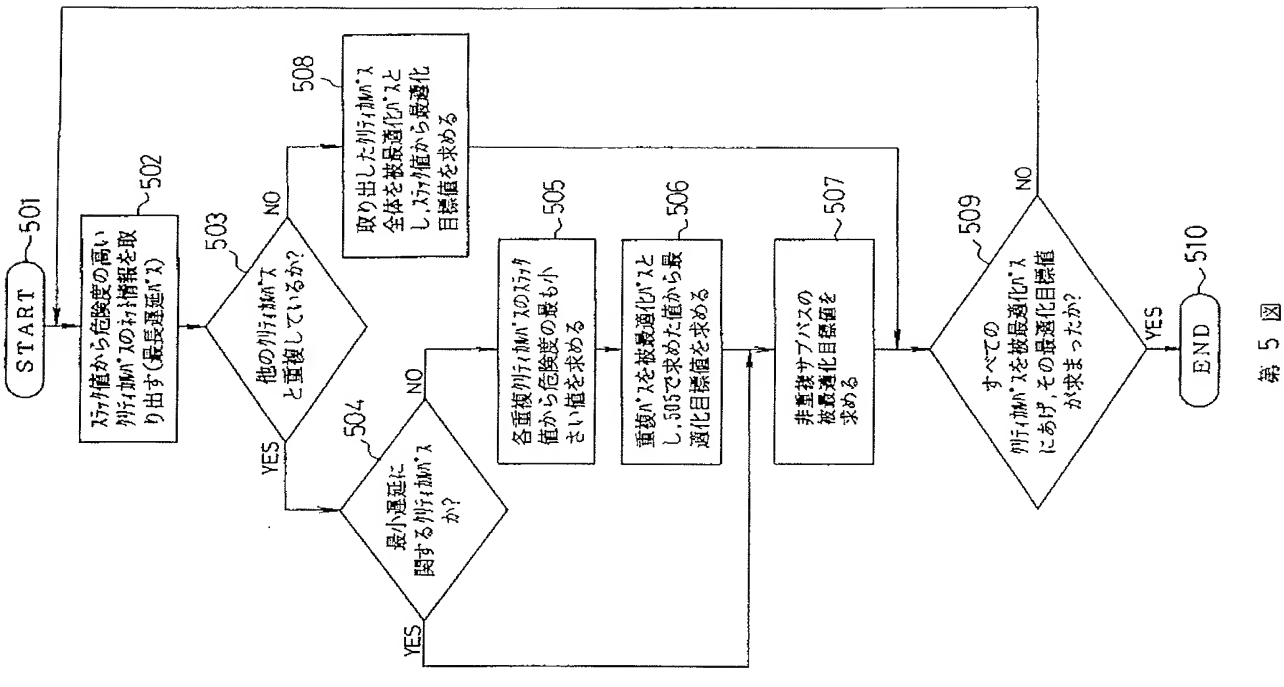
第 3 図


START
401
被最適化パスと最適化目標値を取り出す
402
被最適化パスから被展開機能素子の候補を探し出す
403
被最適化パスに被展開機能素子があるか?
NO
YES
最適化目標値が達成されたか?
NO
YES
409
展開後の機能素子同士または,被最適化パスの端点の機能素子間で最適化を行う
404
被展開機能素子と実際に展開した後の機能回路の遅延段数が最適化目標値に近づくか?
NO
YES
410
被最適化パスが残っているか?
YES
NO
411
展開後の回路のクリティカルパス探索を行う
選択された機能素子を展開する
405
展開後の個々の機能素子に遅延段数を割り付ける
406
展開された部分回路の段数計算を行い,最大最小段数を計算する
407
展開前の機能素子の遅延段数と展開後の部分回路の遅延段数を比較する
408
展開最適化前のクリティカルパスは解消されたか?
YES
NO
新たにクリティカルパスが発生したか?
NO
YES
412
402からの処理へ戻る
最適化目標値が達成されたか?
NO
YES
被最適化パスが残っているか?
YES
NO
最適化対象以外の機能素子の展開最適化を行う
413
END
414

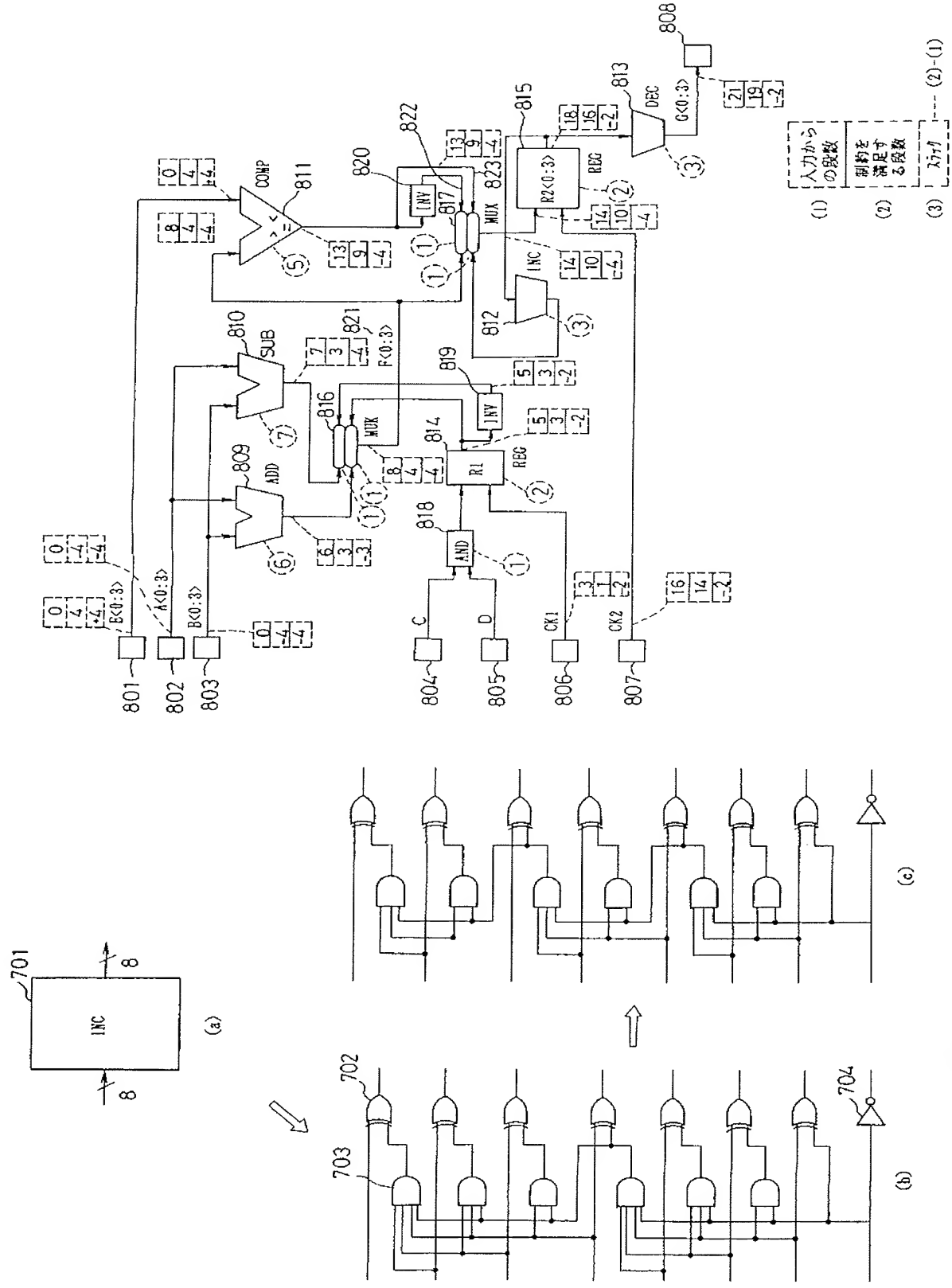
第 4 図


600
CO
S<0:9>
A<0:9>
B<0:9>
CI
(a) 10ビット2入力加算器
606
607
608
(c) 変換例2
601
602
603
604
605
609
(b) 変換例1

第 6 図


START
501
スラック値から危険度の高い
冗行加ﾒﾝﾊﾞｰの行情報を取
り出す(最長遅延ﾊﾟｽ)
502
他の冗行加ﾊﾟｽ
と重複しているか?
503
NO
YES
取り出した冗行加ﾊﾟｽ
全体を被最適化ﾊﾟｽとし,スラック値から最適化
目標値を求める
508
是小遅延に
関する冗行加ﾊﾟｽ
か?
504
NO
YES
各重複冗行加ﾊﾟｽのスラック
値から危険度の最も小
さい値を求める
505
重複ﾊﾟｽを被最適化ﾊﾟｽとし,505で求めた値から最
適化目標値を求める
506
非重複サブパスの
被最適化目標値を
求める
507
すべての
冗行加ﾊﾟｽを被最適化ﾊﾟｽ
にあげ,その最適化目標値
が求まったか?
509
NO
YES
END
510

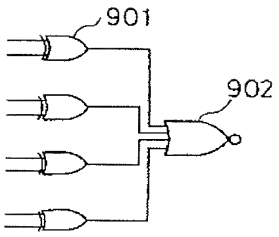
第 5 図


701
INC
8
8
(a)
702
703
704
(b)
(c)

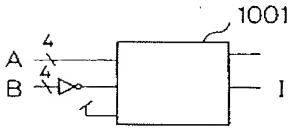
第 7 図

801
802
803
804
805
806
807
808
A<0:3>
B<0:3>
C
D
CK1
CK2
Q<0:3>
COMP
SUB
ADD
INV
MUX
REG
R1
R2<0:3>
AND
INC
DEC
F<0:3>
(1) 入力からの段数
(2) 制約を満足する段数
(3) カット (2)-(1)

第 8 図



第 9 図



第10 図